

NAME

DATE

PERIOD

UNIT 4 · TOPIC 4.8 · WORKSHEET

Scoot Over

Add, insert, replace, and remove, and keep track of who moved where.

About 30 minutes

10 questions

Cumulative through 4.8

TOPIC

4.8



DIRECTIONS

Work each puzzle in pencil. When code prints, write exactly what it prints; a printed list looks like [Fern, Moss]. Every program starts with `import java.util.ArrayList;`, even where it isn't shown. Some puzzles include pieces you won't need.

PART A · 2 QUESTIONS

A Warm up your brain

1 YOU'RE THE JVM

You are the Java Virtual Machine. Run this code in your head. What prints?

```
ArrayList<String> snacks = new ArrayList<String>();
snacks.add("fig");
snacks.add("kiwi");
snacks.add("pear");
String old = snacks.set(1, "plum");
String gone = snacks.remove(0);
System.out.println(old + " " + gone + " " + snacks);
```

A kiwi fig [plum, pear]

B plum fig [plum, pear]

C true true [plum, pear]

D kiwi fig [fig, plum, pear]



THINK FIRST

A list holds 3 frogs. You insert a new frog at index 1. Which frogs get a new index, and which stay put? Guess in the margin; question 2 settles it.

2

PENCIL CHECK

The list starts empty: `ArrayList<String> pond = new ArrayList<String>();`. The calls run in order. After each one, write the whole list and its `size()`.

Call	pond afterward	size()
<code>pond.add("Fern");</code>		
<code>pond.add("Reed");</code>		
<code>pond.add(1, "Moss");</code>		
<code>pond.add(0, "Lily");</code>		
<code>pond.set(2, "Pip");</code>		
<code>pond.remove(1);</code>		



WATCH OUT

`ArrayList` lives in the `java.util` package, not `java.lang`. Leave off `import java.util.ArrayList;` and the compiler stops at every use of the class with `cannot find symbol`.

PART B · 4 PUZZLES

B Puzzles

3 CODE TILES

Hopper's carpool rotates: the frog at the front drives today, then goes to the back of the line. Put the tiles in order so the program prints the output below. Two tiles are decoys and aren't used.

```
ArrayList<String> carpool = new ArrayList<String>();
carpool.add("Fern");
carpool.add("Moss");
carpool.add("Reed");
```

```
System.out.println(carpool);
```

```
carpool.add(driver);
```

```
String driver = carpool.get(0);
```

```
System.out.println(driver + " drives, " + carpool.size() + " waiting");
```

```
carpool.add(0, driver);
```

```
String driver = carpool.remove(0);
```

Target output

```
Fern drives, 2 waiting
```

```
[Moss, Reed, Fern]
```

YOUR PROGRAM

4 SNIPPET POOL

Fill each numbered hole with a snippet from the pool so the program prints [Fern , Moss , Lily , Reed] 4. Each snippet is used at most once.

```
ArrayList<String> seats = new ArrayList<String>();
seats.add("Fern");
seats.add("Reed");
seats. (1) ;
seats. (2) ;
System.out.println(seats + " " + (3) );
```

THE POOL

Each snippet can be used only once, and not every snippet is needed.

add(1, "Moss")	set(1, "Moss")	add(0, "Moss")	add(2, "Lily")
add("Lily")	set(2, "Lily")	seats.size()	seats.length



FAIR QUESTION

If an ArrayList can grow, why would anyone still use an array?

An array is simpler when the number of elements is fixed and known, like the 7 days of a week, and it can hold `int` and `double` values directly. An `ArrayList` is the better fit when elements come and go while the program runs, like names on a waiting list.

5 FACE-OFF

Tonight: `set` and `add(index, obj)` argue about who's better at index 1. Fill in the blanks.

SET	Give me an index and an element, and I _____ what's there. Size: _____.
ADD	I _____. Everyone from that index on scoots _____, and size goes up by _____.
SET	I also hand back the element I _____, in case you want it.
ADD	My index can be as big as <code>size()</code> ; that adds at the _____.
SET	Mine stops at _____. At <code>size()</code> I throw an _____.

6 YOU'RE THE COMPILER

You are the compiler, and then the JVM. For each snippet, decide whether it compiles. If it doesn't, say why. If it does, write what it prints, or name the exception if it crashes.

In 2 to 5, `nums` is a new, empty `ArrayList<Integer>` and `words` is a new, empty `ArrayList<String>`.

Snippet	Compiles?	Output, crash, or why not
1 <code>ArrayList<int> counts;</code>		
2 <code>nums.add(7);</code> <code>System.out.println(nums.get(0) + 1);</code>		
3 <code>words.add(5);</code>		
4 <code>words.add("hi");</code> <code>System.out.println(words.get(1));</code>		
5 <code>System.out.println(words.length);</code>		

PART C · 4 QUESTIONS

C Catch the mistake

7 TWO TRUTHS AND A LIE

Two of these claims about the code are true and one is a lie. Circle the lie, then rewrite it so it's true.

```

1 ArrayList<Integer> scores = new ArrayList<Integer>();
2 scores.add(10);
3 scores.add(20);
4 scores.add(30);
5 Integer out = scores.remove(1);

```

- 1 After line 5, `out` holds 20.
- 2 After line 5, `scores.get(1)` returns 30.
- 3 To remove the 30 instead, you could change line 5 to `scores.remove(30);`.

THE LIE, REWRITTEN

8 ONE-TOKEN FIX

Hopper wants to send Moss and Lily home, leaving [Fern, Reed, Pip]. The program prints [Fern, Reed, Lily] instead. Change exactly one token to fix it.

```
1 ArrayList<String> pond = new ArrayList<String>();
2 pond.add("Fern");
3 pond.add("Moss");
4 pond.add("Reed");
5 pond.add("Lily");
6 pond.add("Pip");
7 pond.remove(1);
8 pond.remove(3);
9 System.out.println(pond);
```

WHAT WENT WRONG, AND YOUR FIX

9 WHO AM I?

ArrayList methods are playing a party game. Name each one, with its parameters.

Clue	I am...
I count the elements. Arrays call their version <code>length</code> .	
I put the new element at the very end and return <code>true</code> .	
I squeeze an element in; everyone after it moves right.	
I hand you the element at an index and leave the list alone.	
I swap in a new element and hand you back the old one.	
I take an element out, close the gap, and hand you what I took.	

10 YOU'RE THE JVM

This time the list holds Frog objects. Use the Frog class above the code. Write the exact output.

Frog.java

```
class Frog {  
    private String name;  
    private int hops;  
    public Frog(String n) {  
        name = n;  
        hops = 0;  
    }  
    public void hop() { hops++; }  
    public int getHops() { return hops; }  
    public String getName() { return name; }  
}
```

In main

```
ArrayList<Frog> pond = new ArrayList<Frog>();  
Frog f = new Frog("Pip");  
pond.add(f);  
f.hop();  
pond.get(0).hop();  
System.out.println(f.getHops());  
f = new Frog("Lily");  
System.out.println(pond.get(0).getName() + " " + pond.size());
```

OUTPUT